PRESENTATION

SMART SYSTEM IN AUTOMATION'S PROJECT TOPIC: IDENTIFY AND MEASURE PRECISELY OBJECT DISTANCE

STUDIO SHODWE

MEMBERS

Huỳnh Du Kiệt
Dương Trọng Anh
Cao Minh Ngọc Anh

2022 PRESENTATION

STUDIO SHODWE

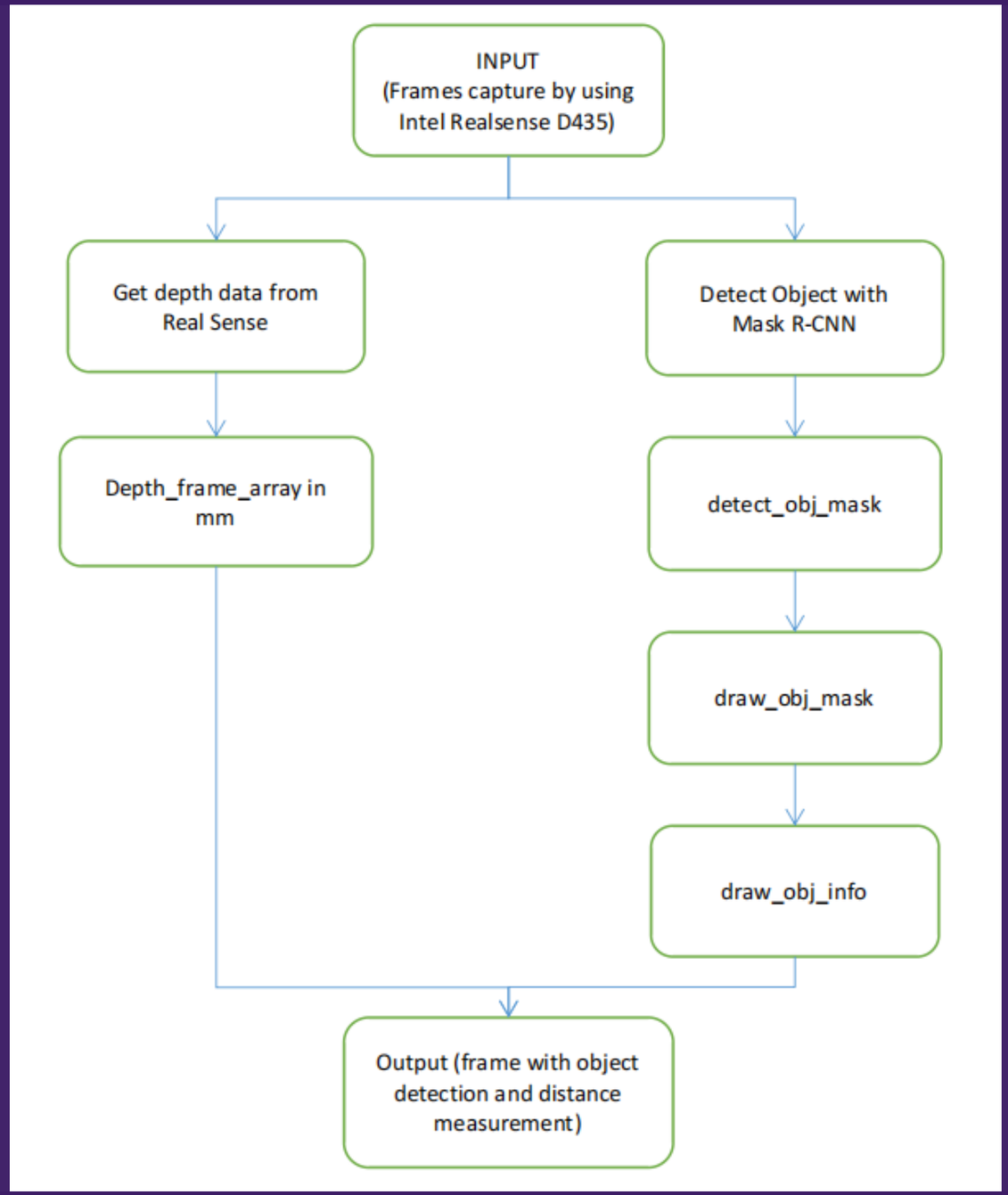
1) INTRODUCTION

a) General information about the Project

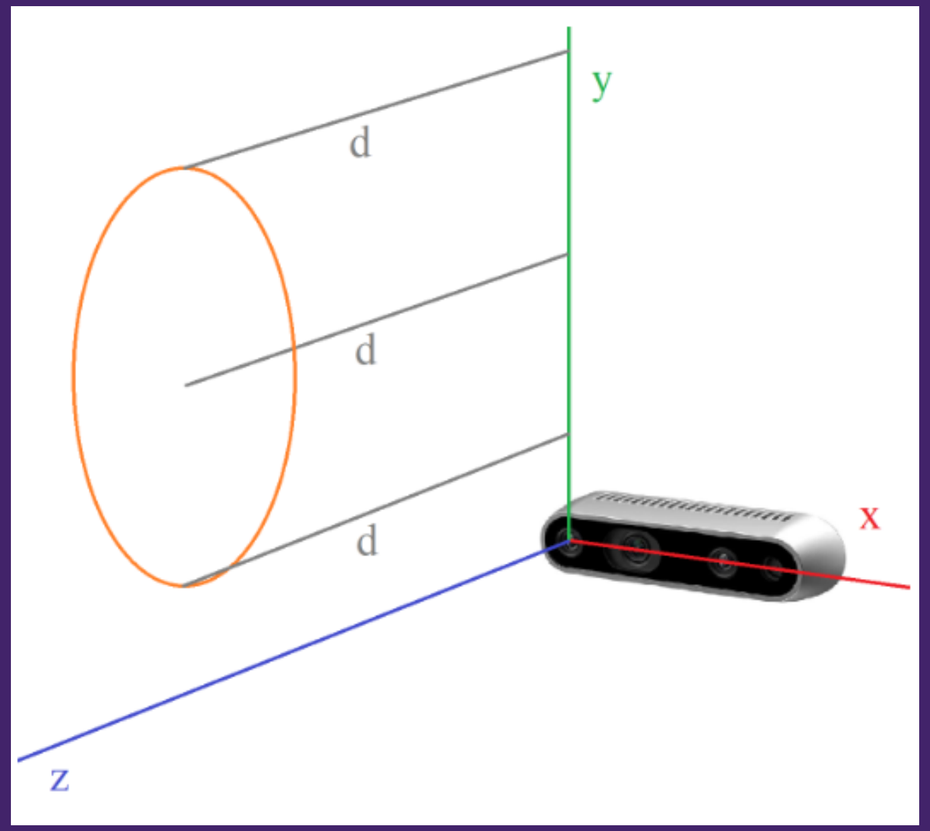
In this project, we will use the depth camera D435 to measure the distance of an object from the camera and also integrate object detection with artificial intelligence into this project.

1) INTRODUCTION

b) Flowchart



STUDIO SHODWE



2) METHODOLOGY

2.1) Measure Object Distance

2.1.1) General information

- The principle of measuring the distance of objects is to display the value of “depth” from the camera lens to the objects. To put it simply, if we put the Intel RealSense D435 on a 3D graph, the camera lens will be on the xy-plane, and the distance will be projection d along the z-axis as shown in the analogy picture

- d will be the value that the camera detects.

2) METHODOLOGY

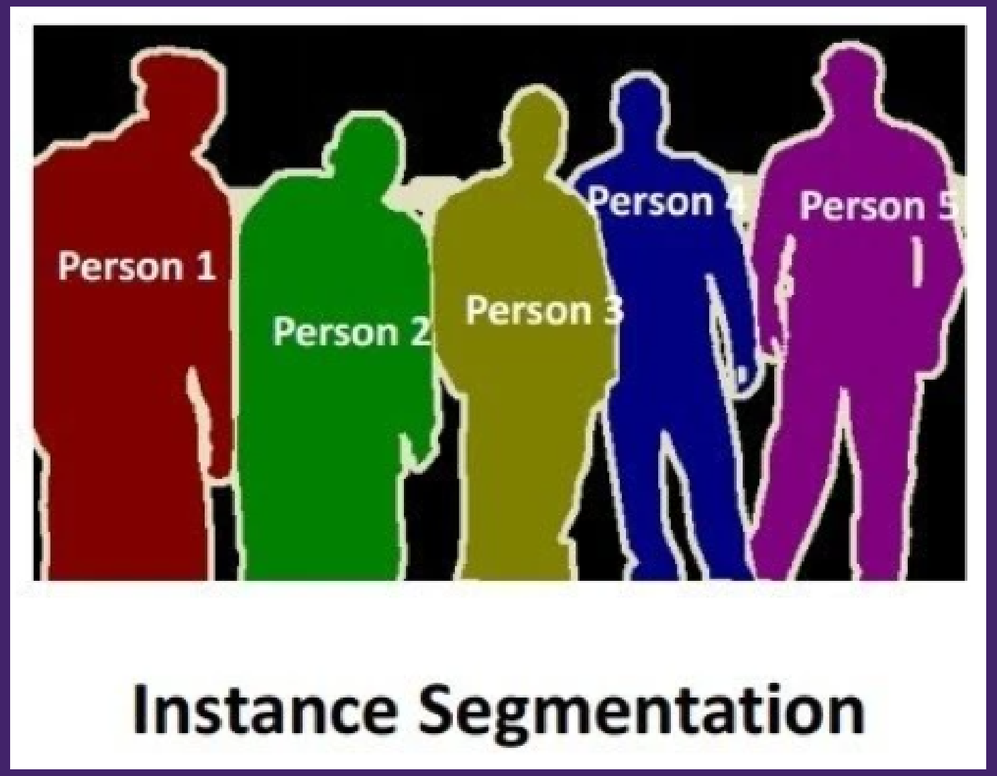
2.1) Measure Object Distance

2.1.2) System Contruction

```
import cv2
from realsense_camera import *
from mask_rcnn import *

# Load Realsense camera
rs = RealsenseCamera()
mrcnn = MaskRCNN()
while True:
    # Get frame in real time from Realsense camera
    ret, bgr_frame, depth_frame = rs.get_frame_stream()
    # Get object mask
    boxes, classes, contours, centers = mrcnn.detect_objects_mask(bgr_frame)
    # Draw object mask
    bgr_frame = mrcnn.draw_object_mask(bgr_frame)
    # Show depth info of the objects
    mrcnn.draw_object_info(bgr_frame, depth_frame)
    cv2.imshow("depth frame", depth_frame)
    cv2.imshow("Bgr frame", bgr_frame)
    key = cv2.waitKey(1)
    if key == 27:
        break
rs.release()
cv2.destroyAllWindows()
```

STUDIO SHODWE



2) METHODOLOGY

2.2) Identify the Object

2.2.1) General information

- In this project, we will use the Mask R-CNN algorithm. Mask R-CNN is an instant segmentation algorithm, which means it can recognize objects in images while concurrently applying a mask to each item. This implies that for a person, you have not just the box but also the coordinates that surround the person.
- Mask R-CNN was invented in 2017, thus it is rather ancient; yet, in the world of computer vision software development, it is still a great method that can be utilized in commercial applications.

2) METHODOLOGY

2.2) Identify the Object

2.2.2a) Initialize function

```
def __init__(self):
        # Loading Mask RCNN
        self.net =
cv2.dnn.readNetFromTensorflow("dnn/frozen_inference_graph_coco.pb","dnn/mask_rcnn_inception_v
2_coco_2018_01_28.pbtxt")
        self.net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)
        self.net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)

        # Generate random colors
        np.random.seed(2)
        self.colors = np.random.randint(0, 255, (90, 3))
        # min=0, max=255, (90,3)= 90 combinations of 3 values
        # Conf threshold
        self.detection_threshold = 0.7
        self.mask_threshold = 0.3
        self.classes = []
        with open("dnn/classes.txt", "r") as file_object:
            for class_name in file_object.readlines():
                class_name = class_name.strip()
                self.classes.append(class_name)
        self.obj_boxes = []
        self.obj_classes = []
        self.obj_centers = []
        self.obj_contours = []
        # Distances
        self.distances = []
```

2) METHODOLOGY

2.2) Identify the Object

2.2.2b) detect_objects_mask function

```
def detect_objects_mask(self, bgr_frame):
    # Convert image into blob: scaling the image down simple to make the image smaller
    # swapping the channel because different framework use different color format
    blob = cv2.dnn.blobFromImage(bgr_frame, swapRB=True)

    self.net.setInput(blob) # put the image into the network

    #access the last layer where we have the information
    #so where we have the boxes and where we have the masks
    boxes, masks = self.net.forward(["detection_out_final", "detection_masks"])
    # Detect objects
    frame_height, frame_width, _ = bgr_frame.shape
    detection_count = boxes.shape[2] #dectection count = maximum object can detect
    # Object Boxes
    self.obj_boxes = []
    self.obj_classes = []
    self.obj_centers = []
    self.obj_contours = []
```

2) METHODOLOGY

2.2) Identify the Object

2.2.2b) detect_objects_mask function(cont)

```
for i in range(detection_count):# loop from 0 to detection_count
    box = boxes[0, 0, i]
    class_id = box[1] # class that object belong to
    score = box[2] # confidence score
    color = self.colors[int(class_id)]
    # if the confidence score is less than 0.7
    # we will not going to display that specific object
    if score < self.detection_threshold:
        continue
    # Get box Coordinates
    x = int(box[3] * frame_width)
    y = int(box[4] * frame_height)
    x2 = int(box[5] * frame_width)
    y2 = int(box[6] * frame_height)
    self.obj_boxes.append([x, y, x2, y2])
    cx = (x + x2) // 2
    cy = (y + y2) // 2
    self.obj_centers.append((cx, cy))
    # append class
    self.obj_classes.append(class_id)
    # Contours
    # Get mask coordinates
    # Get the mask
    mask = masks[i, int(class_id)]
    roi_height, roi_width = y2 - y, x2 - x # size of the object1
```

2) METHODOLOGY

2.2) Identify the Object

2.2.2b)detect_objects_mask function (cont)

```
    # resize the size of mask with the size of the object
     mask = cv2.resize(mask, (roi_width, roi_height))
     # if the confidence is greater than 0.3 we will mask it with a white pixel which value is 255
      # (object we want to detect)
     # if the confidence is less than 0.3 we will mask it with black pixel (background)
     _, mask = cv2.threshold(mask, self.mask_threshold, 255, cv2.THRESH_BINARY)
     # cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE with this two commands we are going to extract the
     # boundaries
     # of the white are on the mask
     contours, _ = cv2.findContours(np.array(mask, np.uint8), cv2.RETR_EXTERNAL,
                                    cv2.CHAIN_APPROX_SIMPLE)
     self.obj_contours.append(contours)
return self.obj_boxes, self.obj_classes, self.obj_contours, self.obj_centers
```

2) METHODOLOGY

2.2) Identify the Object

2.2.2c) draw_object_mask function

```
def draw_object_mask(self, bgr_frame):
      # loop through the detection
      for box, class_id, contours in zip(self.obj_boxes, self.obj_classes, self.obj_contours):
          x, y, x2, y2 = box
          roi = bgr_frame[y: y2, x: x2]
          roi_height, roi_width, _ = roi.shape
          color = self.colors[int(class_id)]

          roi_copy = np.zeros_like(roi)
          for cnt in contours:
              # cv2.f(roi, [cnt], (int(color[0]), int(color[1]), int(color[2])))
              cv2.drawContours(roi, [cnt], - 1, (int(color[0]), int(color[1]), int(color[2])), 3)
              # draw a polygon on the region of interest (roi)
              cv2.fillPoly(roi_copy, [cnt], (int(color[0]), int(color[1]), int(color[2])))
              roi = cv2.addWeighted(roi, 1, roi_copy, 0.5, 0.0)
              bgr_frame[y: y2, x: x2] = roi
      return bgr_frame
```

2) METHODOLOGY

2.2) Identify the Object

2.2.2d) draw_object_info function

```
def draw_object_info(self, bgr_frame, depth_frame):
      # loop through the detection
      for box, class_id, obj_center in zip(self.obj_boxes, self.obj_classes, self.obj_centers):
         x, y, x2, y2 = box

         color = self.colors[int(class_id)]
         color = (int(color[0]), int(color[1]), int(color[2]))
         cx, cy = obj_center
         depth_mm = depth_frame[cy, cx]
         cv2.line(bgr_frame, (cx, y), (cx, y2), color, 1)
         cv2.line(bgr_frame, (x, cy), (x2, cy), color, 1)
         class_name = self.classes[int(class_id)]
         cv2.rectangle(bgr_frame, (x, y), (x + 250, y + 70), color, -1)
         cv2.putText(bgr_frame, class_name.capitalize(), (x + 5, y + 25), 0, 0.8, (255, 255, 255), 2)
         cv2.putText(bgr_frame, "{} cm".format(depth_mm / 10), (x + 5, y + 60), 0, 1.0, (255, 255, 255),2)
         cv2.rectangle(bgr_frame, (x, y), (x2, y2), color, 1)
      return bgr_frame
```

WEAKNESSES AND STRENGTHS

WEAKNESSES

- When we tested the project, there are still some vibrations in the distance measurement (± 2%)
- It takes a few seconds (10-15 seconds) for the object detection function to determine the object correctly.

STRENGTHS

- Although there is a weakness in distance measurement, it measures the distance quite fast and is trustable.
- The object detection function works well since it can detect correctly almost any object that includes in the object's classes

STUDIO SHODWE

DEMONSTATION ON REAL-HARDWARE

STUDIO SHODWE

THANK YOU